The embodiments of the invention in which an exclusive right or privilege is claimed are described as follows:

1. A computer program product for a computer programming environment supporting virtual function calls and supporting both interpretation of functions in a set of functions and execution of compiled code representing functions in the set of functions, the set of functions being referenced in one or more loaded classes in a set of computer code, the computer program product comprising a computer usable medium having computer readable code means embodied in said medium, comprising

    computer readable program code means for generating, for each loaded class:

        a first virtual function table for access by an interpreter for interpreting a call in the computer code to one of the functions in the set of functions, and

        a second virtual function table for access during execution of the compiled code for calling one of the functions in the set of functions.

2. The computer program product of claim 1, wherein

    the first virtual function table comprises interpretation entries, each interpretation entry being associated with a function in the set of functions and pointing to a corresponding function data structure, and

    the second virtual function table comprises compilation entries, each compilation entry being associated with a function in the set of functions and pointing to either a corresponding block of executable code or to a corresponding block of interpreter transition code.

3. The computer program product of claim 2 in which the interpreter transition code corresponding to a compilation entry for a selected associated function is executable to access the function data structure pointed to by the interpretation entry for the said selected associated function.

4. The computer program product of claim 3 in which each function data structure comprises a target address, the target address pointing to either a send target or a compiled transition target, the send target being the address loaded by the interpreter for interpretation of a function in the set of functions, and the compiled transition target being the address for a code block to permit transition to executable code corresponding to a function in the set of functions.

5. The computer program product of claim 1 further comprising, for each loaded class, a class object comprising a first end and a second end, in which the first virtual function table for the loaded class is contiguous with the first end of the class object, and the second virtual function table for the loaded class is contiguous with the second end of the class object and in which the first virtual function table and the second virtual function table are structured symmetrically about the class object.

6. The computer program product of claim 5, wherein

the first virtual function table comprises interpretation entries, each interpretation entry being associated with a function in the set of functions and pointing to a corresponding function data structure and

the second virtual function table comprises compilation entries, each compilation entry being associated with a function in the set of functions and pointing to either a corresponding block of executable code or to a corresponding block of interpreter transition code.

7. The computer program product of claim 6 in which the interpreter transition code corresponding to a compilation entry for a selected associated function is executable to access the function data structure pointed to by the interpretation entry for the said selected associated function.

8. The computer program product of claim 7 in which the interpreter transition code is defined to

access a selected function data structure by calculating an interpretation entry location in the first virtual function table using the symmetrical structure of the first and the second virtual function tables.

9. The computer program product of claim 7, wherein each function data structure comprises a target address, the target address pointing to either a send target or a compiled transition target, the send target being the address loaded by the interpreter for interpretation of a function in the set of functions, and the compiled transition target being the address for a code block to permit transition to executable code corresponding to a function in the set of functions.

10. The computer program product of claim 8, wherein each function data structure comprises a target address, the target address pointing to either a send target or a compiled transition target, the send target being the address loaded by the interpreter for interpretation of a function in the set of functions, and the compiled transition target being the address for a code block to permit transition to executable code corresponding to a function in the set of functions.

11. The computer program product of claim 2 in which each function data structure comprises a counter for determining the timing of compilation of the function associated with the interpretation entry for the function data structure.

12. The computer program product of claim 11 in which the counter is initialized to a predetermined odd value and is decremented by two on each access of the function data structure until the counter reaches a negative value, the counter being replaced with an even-value address on compilation of the function associated with the interpretation entry for the function data structure.

13. The computer program product of claim 6 in which each function data structure comprises a counter for determining the timing of compilation of the function associated with the interpretation entry for the function data structure.

14. The computer program product of claim 13 in which the counter is initialized to a predetermined odd value and is decremented by two on each access of the function data structure until the counter reaches a negative value, the counter being replaced with an even-value address on compilation of the function associated with the interpretation entry for the function data structure.

15. The computer program product of claim 1 further comprising, for each loaded class, a class object and in which the first virtual function table for the loaded class and the second virtual function table for the loaded class are interleaved with each other and are contiguous with the class object.

16. A Java virtual machine comprising an interpreter, supporting virtual method calls and supporting both interpretation of methods in a set of methods and execution of compiled code representing methods in the set of methods, the set of methods being referenced in one or more loaded Java classes, the Java virtual machine comprising a computer usable medium having computer readable code means embodied in said medium, comprising

computer readable program code means for generating, for each loaded Java class:

a first virtual function table for access by the interpreter for interpreting a call to one of the methods and comprising interpretation entries, each interpretation entry being associated with a method and pointing to a corresponding function data structure; and

a second virtual function table for access in the execution of the compiled code to execute a call to one of the methods and comprising compilation entries, each compilation entry being associated with a function in the set of functions and

pointing to either a corresponding block of executable code or to a corresponding block of interpreter transition code,

the interpreter transition code corresponding to a compilation entry for a selected associated function being executable to access the function data structure pointed to by the interpretation entry for the said selected associated function.

17. The Java virtual machine of claim 16 further comprising, for each loaded Java class, a class object comprising a first end and a second end, in which the first virtual function table for the loaded Java class is contiguous with the first end of the class object, and the second virtual function table for the loaded class is contiguous with the second end of the class object and in which the first virtual function table and the second virtual function table are structured symmetrically about the class object.

18. The Java virtual machine of claim 17 in which the interpreter transition code corresponding to a compilation entry for a selected associated function is executable to access the function data structure pointed to by the interpretation entry for the said selected associated function by calculating an interpretation entry location in the first virtual function table using the symmetrical structure of the first and the second virtual function tables.

19. The Java virtual machine of claim 18 in which each function data structure comprises a target address, the target address pointing to either a send target or a compiled transition target, the send target being the address loaded by the interpreter for interpretation of a function in the set of functions, and the compiled transition target being the address for a code block to permit transition to executable code corresponding to a function in the set of functions.

20. A Java language programming environment, supporting virtual method calls and supporting both interpretation of methods in a set of methods and execution of compiled code representing

methods in the set of methods, the set of methods being referenced in one or more loaded Java classes, the Java language programming environment comprising, for each loaded Java class:

a first virtual function table for access by the interpreter for interpreting a call to one of the methods and comprising interpretation entries, each interpretation entry being associated with a method and pointing to a corresponding function data structure; and

a second virtual function table for access in the execution of the compiled code to execute a call to one of the methods and comprising compilation entries, each compilation entry being associated with a function in the set of functions and pointing to either a corresponding block of executable code or to a corresponding block of interpreter transition code,

the interpreter transition code corresponding to a compilation entry for a selected associated function being executable to access the function data structure pointed to by the interpretation entry for the said selected associated function.

21. A method for creating a hybrid application for execution by a computer, said hybrid application comprising interpreted code and compiled code, said hybrid application comprising a function, said method comprising:

creating a first function table for access by an interpreter for interpreting a call in said interpreted code to said function; and

creating a second function table for access during execution of said compiled code, said access for a call in said compiled code to said function.